Creating a Modular Integration Layer in Azure

**DESCRIPTION**

The solution will involve building a **modular integration layer** on **AWS**, which engineers can use to deploy different components of the system.

The solution should be highly configurable, ensuring that users can choose which components to deploy, decide on the authentication method, and ensure that security, monitoring, and IAM (Identity and Access Management) are part of the setup.

**Terraform** will be used to define and provision the infrastructure in an automated and repeatable manner.

The system will consist of modular Terraform configurations that allow for the deployment of different Azure services (Azure Integration Services). Each module will be reusable, making it possible to deploy specific components independently.

The solution should also allow for custom authentication methods, such as using Azure Active Directory (AAD) or service principal credentials.

**MAIN TASKS**

* Research and Design:

1. Research Azure Services: Understand the AWS services needed (e.g., AWS Kubernetes Service, AWS Lambda Functions, EC2 Virtual Machines, etc.).
2. Understand Security Best Practices: Research IAM, encryption, logging, monitoring, and security best practices on AWS.
3. Terraform Basics: Ensure understanding of how to write, plan, and apply Terraform configurations.
4. Define Modules **(\*)**: Design the modular system. Define which components can be modularized, and how users will interact with them (inputs/outputs).
5. Authentication and Security **(\*)**: Research and design authentication mechanisms for different users
6. Monitoring: Research how to incorporate monitoring into the solution (AWS CloudWatch, CloudTrail, etc)

* Develop Modular Components **(\*)** :

1. Terraform Modules: Start developing the core Terraform modules for the integration layer, ensuring they can be used independently and are easily configurable.

Examples of components could include: Network setup (VPCs, Subnets, Security Groups), Compute services (EC2, EKS, etc), Storage solutions (S3, Databases), Security components (AWS WAF, etc )

1. Authentication and IAM: Implement IAM configurations that enable role-based access control (RBAC) and include security policies, JWT Tokens.
2. Logging & Monitoring: Integrate Azure monitoring and alerting systems to track the performance and security status of the deployed resources.

* Build Documentation and Examples:

1. Create Comprehensive Documentation: Document the solution, explaining how to use each module, what variables are required, how to customize deployments, and how to integrate authentication methods.
2. Create Examples: Provide example Terraform scripts for common use cases (e.g., deploying AWS Lambda Functions, setting up a VPC with a secure database).
3. Security Guidelines: Provide guidelines on how to securely configure and use the solution (e.g., use of IAM Roles, AWS KMS, etc.).

* Testing and Validation:

1. Write Automated Tests: Create tests to validate that the Terraform scripts correctly provision resources in AWS.
2. Test Security and IAM Configurations: Test the implemented IAM roles and ensure users have the correct access.
3. Test Logging & Monitoring: Verify that monitoring and logging are correctly set up and provide valuable insights into the deployed resources.
4. User Acceptance Testing: Have internal engineers test the solution to ensure usability and functionality.

* Final Presentation and Delivery **(\*)**

**(\*) Required**

**KEY DELIVERABLES**

* Terraform modules (modular and reusable Terraform code for different AWS components (e.g., network, compute, security, storage).
* Documentation

1. Instructions on how to use the solution.
2. Detailed descriptions of each module.
3. Examples for deploying different components.

**TOOLS**

* AWS
* Terraform
* TFSEC / Checkov
* Visual Studio Code
* GIT

**MINIMUM SUCCESS CRITERIA**

1. Terraform modules to deploy an integration layer
2. Basic script to deploy Terraform to AWS
3. Architecture documentation